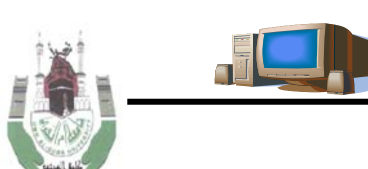
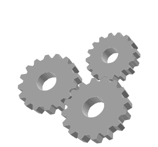


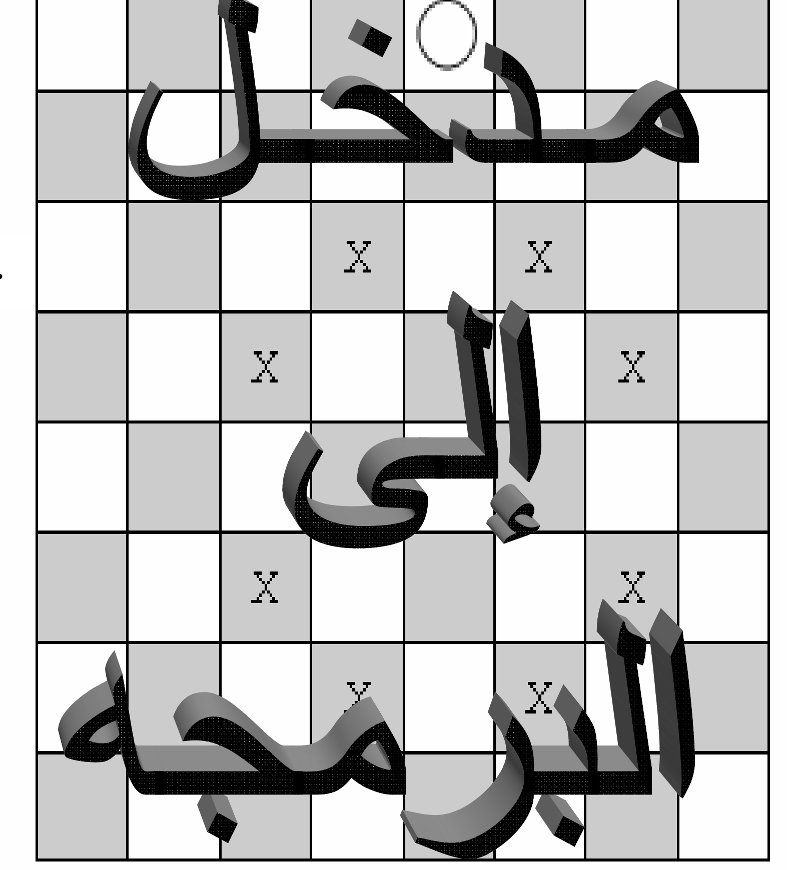


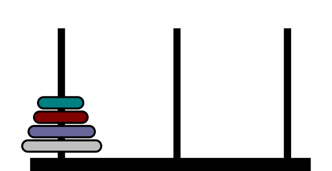


جامعة أم القرى
كلية المجتمع بالباحة
قسم العلوم الهندسيه وتقنياتها
شعبة الحاسب الآلي



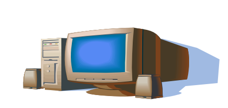
Prog…

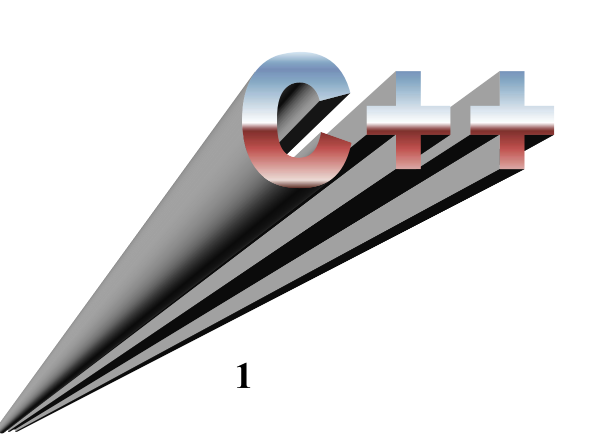












اعداد
أ.هشام حسن

تمهيد

يتكون الحاسب بصورة أساسية من المكونات المادية Hardware والمكونات البرمجية Software والتي تتألف من نظم تشغيل وبرامج تطبيقية، الأداة المستخدمة لإنتاج هذه المكونات هي لغات البرمجة.

وهذه بعض الأمثلة على البرمجيات:

1. نظم قواعد البيانات.
2. النظم الخبيرة
3. نظم التحكم المبرمج
4. نظم التشغيل
5. برامج الحواسب الشخصية كالبرمجيات المكتبية والألعاب
6. برمجة شبكات الحاسب
7. برمجة صفحات الإنترنت

يمكن تقسيم مستويات لغات البرمجة إلى ثلاث مستويات كالآتي:

1. لغة المستوى المنخفض أو ما يعرف بلغة الآلة **Machine Language** وهي لغة الصفر والواحد حيث يمثل الصفر الجهد الكهربي المنخفض والواحد الجهد الكهربي العالي.
2. لغة التجميع **Assembly Language** وتحتوي على أوامر مقتبسة من اللغة الإنجليزية مثل add, . sub
3. لغات المستوى العالي وهي الأقرب لمستوى المبرمج وتحتوي أيضا على أوامر مقتبسة من اللغة الإنجليزية، وهي اللغات التي يستخدمها المبرمجون في تنفيذ التطبيقات المختلفة، بعض هذه اللغات عامة الغرض مثل لغات basic, c, c+....وبعضها خاص الغرض مثل Cobol, simula, prolog ....

تعريف :

البرنامج هو مجموعة من الأوامر والتركيبات البرمجية والتي تتم ترجمتها إلى لغة الآلة ثم تنفيذها.

كل التطبيقات التي تقوم بها على الحاسب ليست إلا تنفيذاً لبرامج كتبت بإحدى لغات البرمجة بعد أن تمت ترجمتها إلى لغة الآلة.

الأمر البرمجي: هو عبارة عن مقطع لغوي يكتب بصيغه محددة لتنفيذ مهمة معينه.

بعض الأمثلة على الأوامر البرمجية:

- أوامر الإدخال وهي التي تسمح للمستخدم إدخال بيانات للحاسب عبر وحدات الإدخال.
- أوامر الإخراج وهي التي تسمح للمستخدم بإظهار البيانات عبر وحدات الإخراج المختلفة.
- الأوامر التي تسمح بفتح الملفات وقراءة بيانات منها أو كتابة بيانات عليها.
- أدوات التحكم في مسار البرنامج من حيث تنفيذ أمر عند توفر شرط محدد أو تكرار تنفيذ أمر معين.

و بالطبع لكل لغة برمجة أسلوبها الخاص في التعبير عن هذه الأوامر.

لغات البرمجة عالية المستوى يمكن تصنيفها إلى ثلاثة تصنيفات من حيث التركيبه العامة كالآتي:

1. **اللغات الإجرائية "procedural languages"** ويتكون البرنامج فيها من عدة أوامر برمجيه متتابعة، ومن أمثلة هذه اللغات fortoran,basic,cobol..........
2. **اللغات الهيكلية "structural languages"** ويكون البرنامج فيها مهيكل أي متكون من تركيبات داخلية تسمى دوال و الدالة تحتوي على الأوامر البر مجيه ، ومن أمثلة هذه اللغات prolog,C,pascal..........
3. **اللغات كائنية التوجه "object oriented languages "** ويتكون البرنامج فيها من تركيبات برمجيه تسمى صفوف تحتوي على دوال و متغيرات، ويتميز هذا النوع بعدة مواصفات برمجيه جديدة كإعادة الاستخدام والوراثة وتغليف الكود والتعددية الشكلية وبالطبع ستتضح هذه المسميات لاحقاً عند دراستها على اعتبار أن اللغة التي سنستخدمها في دراسة هذه المادة من هذا النوع، ومن أمثلة هذه اللغات Java,C++....

يعتبر هذا التصنيف للغات العالية متتابع زمنياً حيث كانت البرمجة الإجرائية هي أولى أساليب البرمجة التي استخدمت ثم ظهرت البرمجة الهيكلية ثم آخرها في الظهور البرمجة الكائنية.

خطوات حل المشكلة وبناء النظام:

لا شك أن هدف البرمجة هو إيجاد الحل لمشكلة ما، ومن ثم بناء نظام يمثل الحل لهذة المشكلة ولهذا الأمر خطوات متتابعة لا يمكن البدء في خطوة إلا إذا انتهينا من التي تسبقها:

1. تعريف المشكلة وجمع البيانات وتحديد المتطلبات، والمقصود بالمتطلبات هنا ماذا نريد من أن نفعل.
2. تحليل النظام وهنا يتم تحديد المؤثرات الأساسية في النظام ودور كل مؤثر وهذا ما يعرف في التحليل الكائني بالـ use-case , كذلك تحديد مخطط الحالة state diagram لكل عنصر من عناصر النظام.

3. تصميم النظام: و هنا يتم بناء المكونات البرمجية للنظام وتخطيطها على الورق باستخدام المخططات الانسيابية flow charts أو الخوارزميات algorithms وكليهما يمثل وصف لخطوات البرنامج غير أن المخططات الإنسيابية تخطيطات بيانيه للتعبير عن هذه الخطوات وسنتناول هذه المخططات مع عرض أمثله مناسبة خلال هذا الفصل إنشاء الله.
4. كتابة الكود: و هنا تكتب أوامر البرنامج بصوره فعلية بإستخدام إحدى لغات البرمجة عالية المستوى.
5. إختبار النظام: ويتم تجريب البرنامج فعلياً وتنفيذة وإكتشاف الأخطاء، ويمكن تصنيف الأخطاء المتوقعة إلى:
   أ. أخطاء في الصيغة syntax error و هي أخطاء في صيغة كتابة أمر معين، مما يؤدي لعدم ترجمة البرنامج وتنفيذة، وبيئة العمل البرمجية غالبا ما تنبه لهذا النوع من الأخطاء وأحياناً تحديد موقع و شكل الخطأ
   ب. أخطاء في تغطية متطلبات النظام، وهذا النوع من الأخطاء لا يمكن اكتشافه بواسطة بيئة العمل البرمجية، ولكن يمكن مراجعة أداء النظام مع المتطلبات التي تم تعريفها في الخطوة الأولى.
6. تدريب الكوادر البشرية على إستخدام النظام الجديد وتطبيق النظام فعلياً على أرض الواقع.

المخططات الإنسيابية flow charts

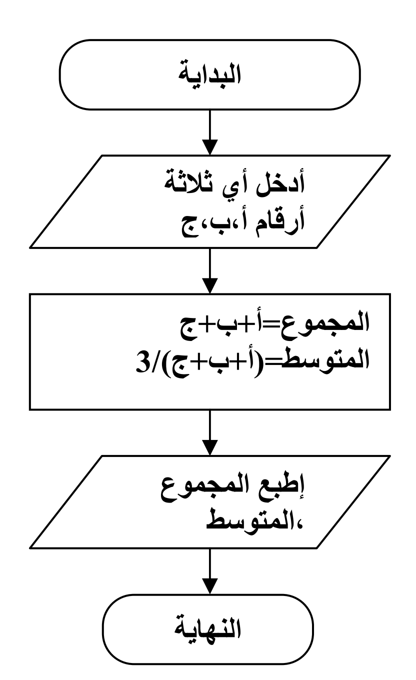
كما ذكرنا من قبل أن هذه المخططات هي وضع خطوات البرنامج في شكل مخططات وتستخدم بعض الرموز المتفق عليها في هذه المخططات لتمثيل الأوامر البرمجية كما هو موضح في الجدول التالي:

الرمز	المدلول
(شكل بيضاوي مستطيل بحواف دائرية)	بداية/ نهاية البرنامج
(مستطيل)	معالجة
(معين)	قرار
(متوازي أضلاع)	إدخال/ إخراج البيانات

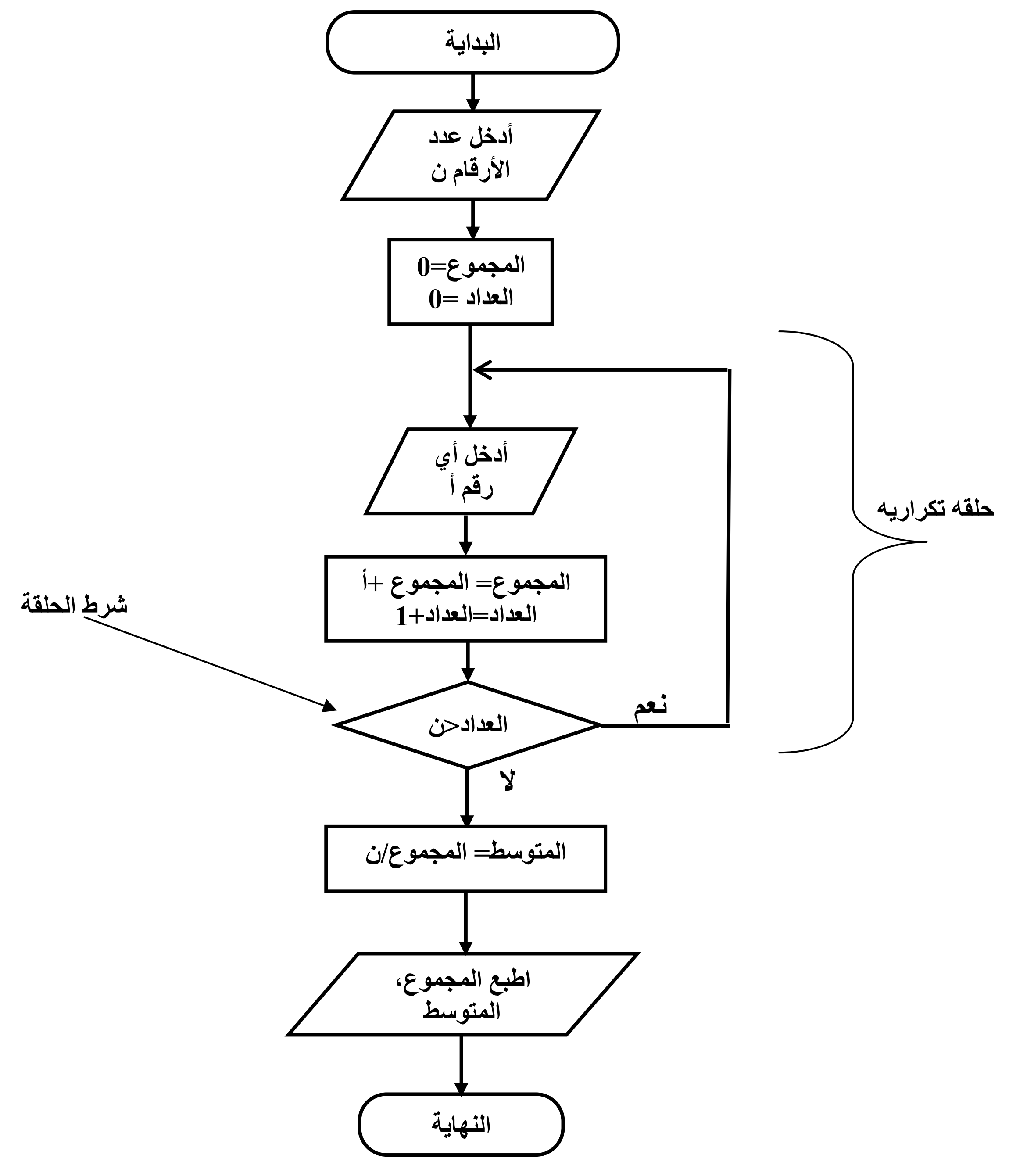
مثال(1.1)
أنشي مخططاً انسيابياً لبرنامج حساب المجموع والمتوسط:
- لأي ثلاثة أرقام مدخلة؟
- لأي عدد من الأرقام

الحل:
- خوارزمية الحل لهذه المشكلة لثلاثة أرقام هي:
  1. أدخل أي 3 أرقام
  2. أحسب المجموع والمتوسط لهذه الأرقام
  3. أطبع المجموع والمتوسط

ويمكن توضيح هذه الخطوات في شكل مخطط إنسيابي:



- لإيجاد الحل لأي عدد من الأرقام نستخدم الحلقات التكرارية أي نكرر تنفيذ أوامر إدخال الرقم وحساب المعادلة لعدد "ن" مرة.



مدخل إلى لغة ++C

تعتبر هذه اللغة من اللغات الكائنية وهي لغة برمجه عامة الغرض فيمكن أن تستخدم للبرامج التطبيقية ونظم التشغيل والتحكم الآلي وبرمجة قواعد البيانات وغيرها من النظم المختلفة.

بنية البرنامج في ++C لابد أن تحتوي على الدالة الرئيسية main ويمكن أن يحتوي البرنامج على دوال أخري سواءً تلك التي ينشئها المبرمج أو من الدوال الجاهزة في مكتبة اللغة المعيارية، وهذه المكتبة تحتوي على عدد هائل من الدوال الجاهزة موجودة داخل ملفات تسمى الملفات الترويسية Header Files فمثلاً ملف الرياضيات المسمى math يحتوي على دوال كالجذر التربيعي واللوغاريتمات والنسب المثلثية والتقريب وغيرها وسيتم الحديث عن الدوال والملفات الترويسية بإذن الله في أبواب لاحقة.

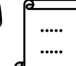
البرنامج التالي يمثل أبسط برنامج في لغة ++C وهو يحتوي علي الحد الأدنى المطلوب وهو الدالة الرئيسية

```
Void main()
{}
```

كلمة void تشير إلى عدم إرجاع الدالة لأي قيمه والأقواس المستديرة عادةً توضع فيها وسائط الدالة وفي هذه الحالة لا توجد وسائط، أما الأقواس المعقوفة { } فهي تمثل بداية ونهاية جسم الدالة الرئيسية.

أوامر إدخال و إخراج البيانات:

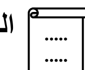
يستخدم التعبير cout للإخراج و cin للإدخال وكلاهما معرفين في ملف الإدخال والإخراج المسمى iostream



الصيغة العامة لإستخدام cout هي

```
cout<<" text";
```

حيث يكتب بين علامتي التنصيص النص المراد إظهارة على الشاشة، أما عند الرغبه في إظهار قيم متغيرات فلا تستخدم علامات التنصيص.



الصيغة العامة لإستخدام cin هي

```
cin>>variable;
```

والمقصود بـ variable المتغير الذي نريد إدخال قيمته عن طريق لوحة المفاتيح.

الصيغه العامه لإدراج ملف في برنامج ما:

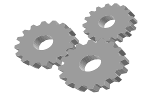
```
#include<file_name>
```

و file_name هو إسم الملف المراد إدراجه ضمن البرنامج

البرنامج الأول في ++C



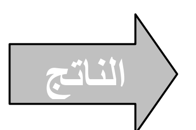
البرنامج التالي يقوم بإظهار نص على شاشة الإظهار



Prog1

```
#include<iostream.h>
void main()

{
cout<<"welcome to C++ world";
}
```



welcome to C++

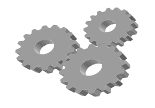
نلاحظ أن ملف الإدخال والإخراج iostream اتخذ الامتداد h، وهذا شأن كل ملفات الترويسه

أدوات الهروب:

هي أدوات تستخدم للتحكم في موقع المخرجات من حيث التحرك الأفقي أو الرأسي عبر السطور حيث تستخدم n\ للنزول سطراً، و t\ للتحرك مسافة أفقية.

أدوات التعليق:

يمكن التعليق على البرنامج لتوضيح عمل داله أو أمر معين لمن يقرأ البرنامج دون أن يؤثر ذلك على عمل البرنامج حيث لا يعتبر التعليق جزءً من البرنامج، تستخدم الأداة // لإضافة تعليق في سطر واحد، بينما يوضع التعليق متعدد الأسطر بين العلامتين /*التعليق*/ .



Prog2

يمكن إعادة كتابة البرنامج السابق مع إضافة أدوات الهروب والتعليق:

```
#include<iostream.h>
void main()

{    //start of main function
cout<<"\twelcome\n to C++ world";
}   /*end of main function
       this a multi_line comment */
```

الناتج

Welcome
to C++ world

<u>تمرين 1</u>

1. أنشي مخططاً انسيابياً لبرنامج يحسب مساحة ومحيط المستطيل عند إدخال الطول و العرض؟
2. أكتب برنامج بلغة ++C يظهر على الشاشة النص:

This is my first program in
C++ language

المتغيرات والثوابت Variables & Constants

المتغير هو موقع في الذاكرة حيث يمكن تخزين القيم واسترجاعها عند الحاجة، سعة الذاكرة التي تخصص للمتغير تعتمد على نوعه، والذي يمكن أن يكون عدداً صحيحاً أو حقيقياً أو متغيراً حرفياً الجدول التالي يوضح أنواع المتغيرات في لغة ++C:

النوع	الحجم	التمثيل	القيم
int	2بايت عدا في ويندوز 95 أو ويندوز NT فحجمه 4 بايت	الأعداد الصحيحة	32768- إلى32767 إلا في حالة 4 بايت فالقيم من 2147483648-إلى2147483647
short int	2 بايت	الأعداد الصحيحة	32768- إلى32767
long int	4 بايت	الأعداد الصحيحة	2147483648- إلى2147483647
float	4 بايت	الأعداد الحقيقية	1.2e-38- إلى3.4e38
double	8 بايت	الأعداد الحقيقية	2.2e-308- إلى 1.8e308
char	1 بايت	الأحرف	256 حرفا " أحرف آسكي "

ملحوظات:

- *يستخدم التعبير* signed و unsigned للمتغيرات الرقمية لتحديد هل تشمل الأرقام الموجبة والسالبة أم لا، فالتعبير unsigned يدل على الأعداد الموجبة فقط مما يعني تخصيص مدى القيم في الأرقام الموجبة فقط، أما signed فتدل على الأعداد الموجبة والسالبة وإذا لم يذكر أي من التعبيرين فيؤخذ على أنه signed.

  مثلاً التعبير **unsigned short int** يمثل من 0 إلى65535 .
  أما التعبير **signed short int** أو **short int** فيمثل الأعداد من 32768- إلى32767.

- الأنواع int, short int, long int جميعها تمثل الأعداد الصحيحة والإختلاف في المدي الذي تمثلة.
- الأنواع float, doubleجميعها تمثل الأعداد الحقيقية والإختلاف في المدي الذي تمثلة.
- Char تمثل كل أحرف آسكي وهي تشمل الحروف اللغوية والعلامات والأقواس والأرقام غير أنها يتعامل معها على أنها أحرف أي لا تخضع للعمليات المنطقية والحسابية.

تعريف المتغيرات:

يتم تعريف المتغير بذكر نوعه ثم اسمه بحيث يفصل بينهما فراغ وكأي سطر برمجي لابد من الانتهاء بفاصلة منقوطة

الصيغة العامة لتعريف المتغيرات variable_type variable_name;

مثلاً التعريف ;int x يخص تعريف متغير إسمه x على أنه عدد صحيح، والتعريف ;float area لعدد حقيقي area.
يمكن تعريف عدة متغيرات في سطر واحد إذا كانت من نوع واحد فالتعريف
;char a,b,c يعني أن a,b,c متغيرات حرفيه.

قيود على إسم المتغير

1. **يجب ألا يحتوي الإسم على فراغ.**
2. **يجب ألا يحتوي الإسم على العلامات الخاصه مثل +،-،*،/.#،@...........**
3. **يجب ألا يبتدئ الإسم برقم.**
4. **يجب ألا يمثل الإسم كلمه من كلمات اللغه المحجوزة وهي الكلمات المستخدمة في الأوامر مثل for, if, else, int, float, long, ...**

ملحوظات:

- يمكن أن يكون الرقم في وسط أو آخر الإسم.
- الكلمات المحجوزة تظهر بتنسيق مميز في معظم المصرفات التي تكتب فيها البرامج.
- هنالك حساسية لحالة الحرف في تسمية المتغير أي أن area مثلاً تختلف عن AREA.

تعيين قيم المتغيرات:

تتم نسبة القيم إلى المتغيرات بإستخدام معامل الإسناد = والذي ينسب القيمة التي علي يمينه للمتغير علي يساره، كما في المثال التالي:

x=5; حيث تنسب القيمة 5 للمتغير x وهذا يعني تخزين هذه القيمة للموقع المحجوز لـ x.

المعاملات الرياضية Mathematical Operators

لكتابة القوانين والمعادلات الخاصة بنظام معين تستخدم هذه المعاملات الرياضية، وهي موضحة في الجدول التالي:

المعامل	العملية الرياضية
+	**الجمع**
-	**الطرح**
*	**الضرب**
/	**القسمة**
%	**باقي القسمة الصحيح مثلاً 23%7=2**

كل هذه الأدوات تعمل في الاتجاه من اليسار إلى اليمين

الأولوية في تنفيذ العمليات الرياضية

تجعل لغة C++ الأولوية الأولى في التنفيذ لعمليات الضرب والقسمة وباقي القسمة الصحيح على السواء ثم عمليتي الجمع والطرح. وإذا احتوى تعبير رياضي على أكثر من عملية لها نفس الأولوية فإن العملية التي على اليسار تنفذ أولاً. وفي كل الأحوال فإن العمليات داخل الأقواس تنفذ أولاً.

مثال(2.1) ما قيمة x في كل مما يأتي:

أ. X=4+2*5

ب. X=5%3*7+2

ج. X=5+(4%3)

الحل:

أ. X=4+10=14

ب. X=2*7+2=14+2=16

ج. X=5+1=6

ملحوظة:

- يمكن استخدام معامل الإسناد= مع أي من المعاملات الرياضية عندما يراد إسناد قيمه لمتغير معين بدلالة نفس المتغير كما هو مبين في الجدول:

التعبير	التعبير المختصر
x=x+3	x+=3
a=a*2	a*=2

مثال(2.2) أكتب برنامجاً بلغة C++ يحسب مساحة ومحيط المستطيل عند إدخال الطول و العرض؟

الحل: في هذا البرنامج نتعامل مع أربعة متغيرات هي الطول **length**، العرض **width**، المساحة **area**، والمحيط **perimeter**. هذه المتغيرات لا يشترط أن تكون أعداداً صحيحة **int** بل يمكن أن تحتوي على كسور لذا فإن من المناسب لها أن تعرف أعداداً حقيقية **float**، والطول والعرض سنحصل على قيمها عن طريق إدخال المستخدم. أما المساحة و المحيط فتحسبان بالعلاقتين المعروفتين:

المساحة=الطول * العرض

المحيط=2*(الطول+العرض)

لذا يكون البرنامج كالآتي:

Prog3

```
#include<iostream.h>
void main()
{
float length,width,area,perimeter;   //variables definition
cout<<"\n\n\tCalculation of the Area & perimeter for the Rectangle  ";
cout<<"\n\tenter the length of the rectangle ";
cin>>length;
cout<<"\n\tenter the width of the rectangle ";
cin>>width;
area=length*width;
perimeter=2*(length+width);
cout<<"\n\tthe area of this rectangle is = "<<area;
cout<<"\n\tthe perimeter of this rectangle is = "<<perimeter;
}
```

المعاملات الخاصة بتغيير قيمة متغير

- **معامل الزيادة بواحد"++"** ويمكن أن يكون قبلي **pre_ incrimination** بحيث تتم الزيادة قبل تنفيذ الأمر الحالي ويكتب المعامل قبل اسم المتغير أو بعدي **post_ incrimination** بحيث تتم الزيادة بعد تنفيذ الأمر الحالي مباشرةً ويكتب المعامل بعد اسم المتغير.
- **معامل النقصان بواحد"--"** ويمكن أن يكون قبلي **pre_ decrimintation** بحيث يكون النقصان قبل تنفيذ الأمر الحالي ويكتب المعامل قبل اسم المتغير أو بعدي **post_ decrimintation** بحيث يتم النقصان بعد تنفيذ الأمر الحالي مباشرةً ويكتب المعامل بعد اسم المتغير.



Prog4

برنامج يوضح الفرق بين "++" و "--":

```
#include<iostream.h>
void main()
{
int  a=5,b=8;
cout<<"\na= "<<++a;
cout<<"\nb= "<<b++;
cout<<"\n after incrementation b= "<<b;
}
```

الناتج

```
a=6
b=8
after incrimination b=9
```

الثوابت **Constants**

الثابت يأخذ نفس الحيز التخزيني للمتغير غير أن قيمته غير قابله للتغير. يتم الإعلان عن المتغير باستخدام الكلمة المحجوزه **const** يعقبها نوع الثابت ثم اسناد قيمه لهذا الثابت كما في البرنامج التالي الذي يحسب مساحة الدائره عند إدخال قيمة نصف قطرها.



Prog5

```
#include<iostream.h>
void main()
{
float radius,area;
const float pi;
cout<<"enter the circle radius:";
cin>>radius;
  area=pi*radius*radius;
  cout<<"\nthe area of a circle is = "<<area;
  }
```

ملحوظات

- **لابد من اسناد قيمة للثابت عند الإعلان عنه.**
- **إذا لم يحدد نوع الثابت فإنه يعتبر افتراضيا عدد صحيح int.**

تمرين 2

1. إذا كان y متغيراً من نوع int فما قيمته في كل مما يأتي:

أ. y=10*7%(5+8-3)

ب. y=18/5+4

2. اكتب برنامجاً لحساب **الدخل الإجمالي total_income** لموظف إذا علم **الراتب الأساسي bs**، **بدل النقل t** "5% من الراتب الأساسي"، **بدل خبره e**، حيث يحسب الدخل الإجمالي كالآتي: **total_income=bs+t+e**

3. في البرنامج التالي توجد ثلاثه أخطاء استخرجها، مع تحديد سبب الخطأ، ثم أعد كتابة البرنامج بعد تصحيح الأخطاء:

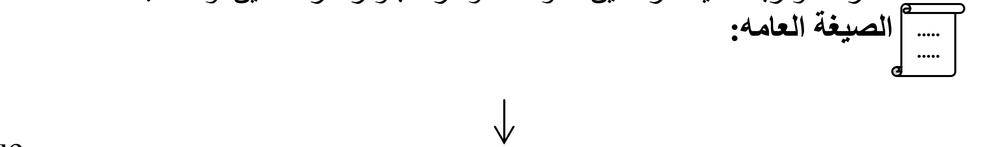
```
void main()
{
int y=6
x=4;
y=y+x;
cout<<y;
}
```

بُنَى التحكم Control Structures

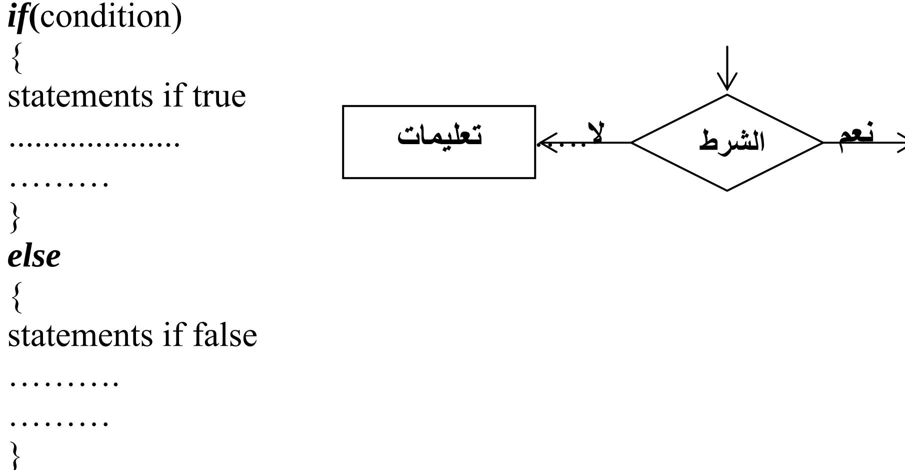
المقصود بذلك التركيبه التي تتحكم في مسار تنفيذ البرنامج، فإلى الآن يتم تنفيذ البرنامج بصورة تتابعيه إلا إذا استخدمت هذه البني، وبنى التحكم تشمل الشرط والحلقات التكرارية.

بنية الشرط في ++C:

الشرط هو ربط تنفيذ أمر معين " أو عدة أوامر " بتوفر شرط معين أوعدمه.



الصيغة العامه:




```
if(condition)
{
statements if true
...................
………
}
else
{
statements if false
……….
………
}
```

نلاحظ من الصيغه العامه استخدام أداة الشرط ifيعقبها الشرط ثم الأمر أو الأوامر المطلوب تنفيذها عند توفر هذا الشرط، لاحظ أن هذه الأوامر لن تنفذ إلا عند توفر الشرط وهذه الأجزاء إجبارية في بنية الشرط " أي يمكن أن نكتفي بالشرط والتعليمات التي تنفذ عند توفره دون طرح خيار آخر. ويمكن استخدام الأداة else لربط الأوامر التي يراد لها أن تنفذ عند انتفاء الشرط.

ملحوظات:

- عندما تكون الأوامر متعددة يجب استخدام الأقواس الحصريه { ، } ويمكن الاستغناء عنها في حالة الأمر الواحد.
- تستخدم المعاملات العلائقية لصياغة الشرط وهي مبينه في جدول لاحق.
- عندما يكون الشرط مركباً تستخدم المعاملات المنطقية للربط بين الشروط.

المعاملات العلائقية Relational Operators

وهي تحدد علاقة متغير بكميه، أو متغير بمتغير آخر وهي ستة تشمل أكبر من، أكبر من أو يساوي، أصغر من، أصغر من أو يساوي، يساوي، لا يساوي وهي موضحه في الجدول التالي.

الأداة	المعنى
>	أكبر من
>=	أكبر من أو يساوي
<	أصغر من
<=	أصغر من أو يساوي
==	يساوي
!=	لا يساوي

كل هذه الأدوات تعمل في الاتجاه من اليسار إلى اليمين

مشكلة الخلط بين = و==

الكثير من الطلاب يخلطون بين معنيي هذين المعاملين، فمعامل الإسناد = يستخدم لنسبة كميه على يمينه إلى متغير على يساره، إذن فهذا المعامل لا يعني يساوي بل يعني أسند قيمه. أما == فهو معامل علائقي يعني يساوي ولا يستخدم إلا مع if.

مثال(1.3) عبر عن الشروط التاليه بصيغة تناسب لغة ++C.

1. إذا كانت x أكبر من y فإن d=5.
2. إذا كانت الدرجة d أكبر أو يساوي 60 اطبع ناجح "pass" وإلا اطبع راسب "fail".
3. إذا كانت قيمة الجذر الربيعي sr أقل من 0 اطبع عدد تخيلي"imaginary number".
4. إذا كانت a لا تساوي 5 فإن b=2a+3، وc=3a-4b، وإلا فإن b=2a-11 ، وc=3a+b.

الحل:

```
1.  if(x>y)
    d=5;
2.  if(d>=60)
        cout<<"pass";
    else
        cout<<"fail";
3.  if(sr<0)
        cout<<" imaginary number";
4.  if(a!=5)
    {
       b=2a+3;
       c=3a-4b;
    }
   else
    {
       b=2a-11;
         c=3a+b;
    }
```

المعاملات المنطقية **Logical Operators**

ويتم عن طريقها الربط بين الشروط وهي ثلاثة معاملات مبينه في الجدول التالي:

الأداة	المعنى	اتجاه التنفيذ
&&	تعني and "و"	كلا الاتجاهين
||	تعني or "أو"	كلا الاتجاهين
!	تعني not لا	اليسار لليمين

ملحوظه:

- **أولوية التنفيذ لهذه المعاملات للتي تكون داخل أقواس وإلا فإن الأولوية تكون من اليسار لليمين.**

جداول الحقيقة Truth Tables

كما هو معلوم أن نتيجة الشرط إما صواب أو خطأtrue or false، وفي حالة الجمله الشرطيه المركبه فإن الناتج يعتمد على الشرطين المكونين للجمله ونوع الرابط المنطقي، وهذا موضح فيما يعرف في علم الرياضيات المتقطعهdiscrete mathematics بجداول الخطأ والصواب أو جداول الحقيقه truth tables والموضحه أدناه. "سنرمز للشرطين بالحرفين A,B وللصواب بالحرفT والخطأ بالحرفF"

أولاً جدول الحقيقه للرابط المنطقي &&

A	B	Result
T	T	T
T	F	F
F	T	F
F	F	F

ثانياً جدول الحقيقه للرابط المنطقي ||

A	B	Result
T	T	T
T	F	T
F	T	T
F	F	F

ثالثاً جدول الحقيقه للرابط المنطقي !

A	!B
T	F
F	T

مثال(2.3) عبر عن الشروط التاليه بصيغة تناسب لغة ++C.

1. إذا كانت x أكبر من y و yتساوي 0 فإن d=5.
2. إذا كانت الدرجة d أكبر أو يساوي 80 أوd أقل من 90 اطبع "B".
3. إذا لم تكن(x أكبر من y و yتساوي 0) أو xتساوي 3فإن dتساوي10 وإلا فإن d تساوي 20.

الحل:

```
1. if(x>y&&y==0)
       d=5;
2. if(d>=80||d<90)
       cout<<"B";
3. if(!(x>y&&y==10)||x==3)
      d=10;
   else
      d=20;
```

خرائط كارنو Carno Maps

في حالة الشروط المعقدة تستخدم خرائط كارنو للوصول للصيغة الصحيحه والمبسطة للشرط، هذه الخرائط عبارة عن تمثيل لكل الاحتمالات الممكنة للشروط ويؤخذ الشرط المركب الذي يؤدي إلى صحة العبارة المنطقيه بمجملها.

مثال(3.3)

في رقعة الشطرنج تتحرك قطعة الحصان بشكل يمثل الحرف L في كل الاتجاهات كما هو مبين بالشكل:

				○			
			X		X		
		X				X	
		X				X	
			X		X		

حيث المواقع الموضحة بـ X تمثل الأماكن الممكن التحرك إليها.

إذا اعتبرنا أن مقدار التحرك الأفقي nxومقدار التحرك الرأسي n_y والموقع الحالي للقطعه (x,y) فإن الشروط الواجب توفرها لتحريك القطعه هي:

A- n_x==1

n_x==2

B- n_y==1

C- n_y==2

D- x+ n_x<=8

E- x- n_x>=1

F- y+ n_y<=8

G- y- n_y>=1

الشروط H,G,F,E لضمان تحرك القطعة ضمن حدود الرقعة وهي تخص تحركات جميع القطع، ويجب توفرها جميعاً في آن واحد أي أن

الرابط فيما بينها هو && أما الشروط D,C,B,A فتخص الحركة في الاتجاه L الروابط بين هذه الشروط يمكن استنتاجها عن طريق خرائط كارنو حيث توضع كل الاحتمالات الممكنة للعلاقات بين الشروط D,C,B,A وعددها 2^4 والعدد 4 يشير إلى عدد الشروط، أما العدد2 يشير إلى احتمالي الصواب والخطأ لكل شرط، في خرائط كارنو يوضع شرطين على كل جانب وتؤخذ الإحتمالات الممكنه لكل منهما كما هو موضح:

	AB	A !B	!A B	!A !B
CD	0	0	0	0
C !D	0	0	1	0
!C D	0	1	0	0
!C !D	0	0	0	0

نلاحظ استخدام 0 للدلاله على عدم إمكانية هذا الاحتمال، واستخدام 1 للدلاله على تحقق الإحتمال، لذا نجد أن الجمله المنطقيه يمكن أن تصاغ بدلالة الحروف D,C,B,A كالآتي:

(A &&!B&&!C&&D)||(!A&&B&&C&&!D)

وبعد إضافة شروط وجود القطعه على الرقعة تكون الجمله المنطقيه:

(E&&F&&G&&H)&&[(A &&!B&&!C&&D)||(!A&&B&&C&&!D)]

ثم بعد ذلك يتم التعويض عن الحروفA,B,C,D,E,F,G,H بالشروط، وبالتالي يكون الشرط كالآتي:

if$[(x+n_x<=8\&\&x-n_x>=1\&\&y+n_y<=8\&\&y-n_y>=1)]\&\&[(n_x==1\&\&n_y==2)||(n_x==2\&\&n_y==1)]$

نلاحظ أنه بالرغم من تعقيد الشرط الناتج إلا أنه أمكن الوصول إلية بصوره سريعة بواسطة خرائط كارنو.

الاختيار المتعدد:

الملاحظ على جملة **if-else** وجود مسارين فقط يجب أن ينفذ أحدهما أما إن كانت المسارات " الخيارات" أكثر من اثنين فتستخدم جملة **switch-case.**

الصيغه العامه

```
switch(variable)
{
 case value1:
statement/statements
break;
case value2:
statement/statements
break;
case value3:
statement/statements
break;
.
.
default:
statement/statements
break;
}
```

لاحظ أن value1، value2، value3 هي قيم للمتغير variable المذكور مع الكلمة المحجوزةswitch ، وتمثل كل case خيار، إذن فقيمة المتغير تحدد الخيار الذي سينفذ "أي تقوم مقام الشرط" وإذا فشلت كل القيم فإن الأوامر التي تلي الكلمة المحجوزة default ستنفذ تلقائياً، نلاحظ أن كل تعليمه ختمت بالكلمة المحجوزة break للإشارة إلى انتهاء التعليمات البرمجيه للخيار المعين.

مثال(3.4)

اكتب برنامجاً بلغة C++ يحسب الدخل الكلي لموظف*total_income* , إذا علمت درجته الوظيفية *grade* حيث يحسب الدخل الكلي بالمعادلة:

total_income=bsaic_salary+bonus+fees

وbsaic_salary هو الراتب الأساسي، *bonus* يمثل العلاوة، و*fees* البدلات وقيم هذه المتغيرات تعتمد على الدرجة الوظيفية حسب الجدول التالي:

grade	bsaic_salary	bonus	fees
1	6000	=.04*bsaic_salary	700
2	5200	=.04*bsaic_salary	700
3	5000	=.03*bsaic_salary	500
4	3000	=.02*bsaic_salary	400
Other grades	2000	=.01*bsaic_salary	200

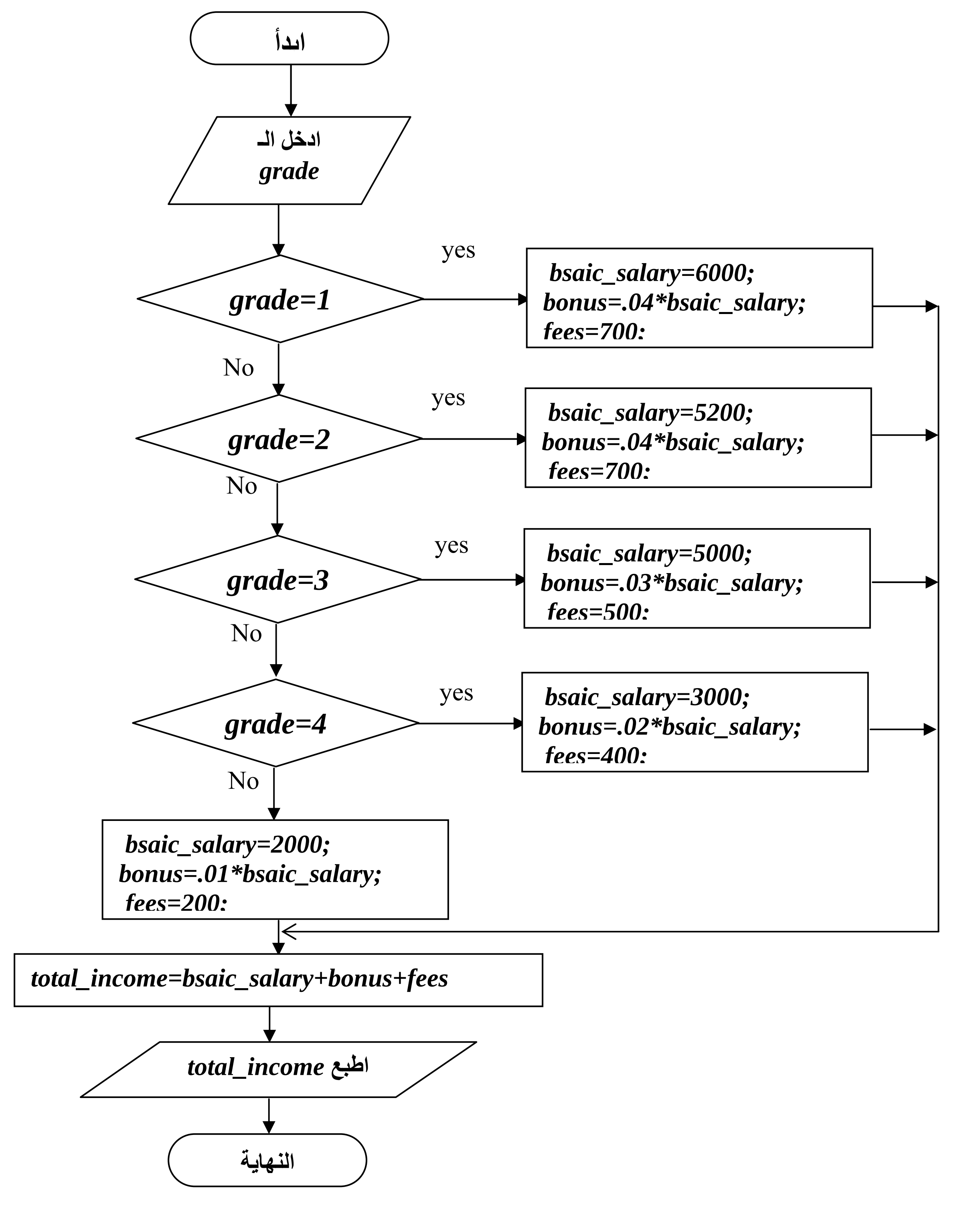
الحل:

نبدأ أولاً بتحديد المتغيرات التي سنتعامل معها

Grade ويتم إدخاله عن طريق المستخدم ويتم تعريفه كعدد صحيح

bsaic_salary، *bonus*، *fees* وهذه المتغيرات تحسب حسب الجدول أعلاه

total_income ويحسب حسب المعادله أعلاه. بالتالي يمكن تصميم المخطط الانسيابي كالآتي:



وبالتالي يكون البرنامج كالآتي:



Prog6

```
#include<iostream.h>
void main()
{
    int grade;
    float bonus, total_income,bsaic_salary,fees;
         cout<<"\n enter the employee grade: ";
         cin>>grade;
  switch(grade)
  {
   case 1:
       bsaic_salary=6000;
      bonus=.04*bsaic_salary;
      fees=700;
 break;
 case 2:
     bsaic_salary=5200;
     bonus=.04*bsaic_salary;
     fees=700;
 break;
case 3:
   bsaic_salary=5000;
bonus=.03*bsaic_salary;
fees=500;
 break;
case 4:
bsaic_salary=3000;
bonus=.02*bsaic_salary;
fees=400;
break;
default:
bsaic_salary=2000;
bonus=.01*bsaic_salary;
fees=200;
 break;
}
total_income=bsaic_salary+bonus+fees;
  cout<<" the total income for this employee is "<<total_income;
  }
```

ملحوظه:

- *عند ارتباط الخيارات بأكثر من متغير، تكتب هذه المتغيرات مع عبارة switch بينها فواصل، وكذلك قيم المتغيرات، تكتب قيمها مع عبارة case بينها فواصل.*

الحلقات التكرارية Loops

تمثل هذه الحلقات الجزء الثاني من بنى التحكم فعند الرغبة في تكرار أمر" أو أوامر" معينه تستخدم هذه الحلقات، و سنتناول في هذا الدرس ثلاثة أنواع من هذه الحلقات. يجب عند تصميم الحلقة تحديد الآتي:

- الأوامر التي تحتاج إلى تكرار لتوضع داخل جسم الحلقة.
- عدد مرات تنفيذ الحلقة، وذلك عن طريق متغير من نوع int يسمى عداد الحلقة وصياغة شرط استمرار الحلقة أو توقفها.

1- حلقة for

الصيغة العامة

```
for(counter_initial_value;condition;counter_incrementation/decrementation)
{
.        statements " loop body"
.
}
```

الصيغة العامة تحتوي على :

for و هي كلمه محجوزة وبين القوسين المستديرين توجد ثلاثة جمل برمجية:

- counter_initial_value و هي الجملة المحتوية على إعطاء عداد الحلقة counter القيمه الإبتدائية، ويمكن أن تحتوي هذه الخطوة على تعريف العداد إذا لم يكن أعلن عن تعريفه من قبل، ويمكن أن تتم عمليتي التعريف وإعطاء القيمه الإبتدائية قبل الحلقة وبالتالي نستغني عن هذا التعبير هنا.
- condition و هو شرط استمرار الحلقة أي أن الحلقه تستمر إذا كان الشرط صحيح وتتوقف إذا انتفى الشرط، و هذا الشرط يكتب باستخدام المعاملات العلائقيه آنفة الذكر، كما يمكن أن يكون الشرط مركباً.
- counter_incrimintation/decrimintation و هو معدل الزيادة incrimination أو النقصان decrimintation في قيمة عداد الحلقة، ويمكن أن تتم هذه الخطوة ضمن الأوامر داخل جسم الحلقة وبالتالي نستغني عن هذا التعبير هنا، ويستخدم لذلك معاملات خاصة كما هو موضح:

x++ ⟺ x=x+1

x-- ⟺ x=x-1

x=*variable* "*mathematical_operator*" " *value*"

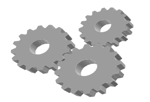
x"*mathematical_operator*"= " *value*"

أمثلة توضيحيه

- **for(int** i=3;i<=8;i++) {…….} تم تعريف العداد وإعطاء القيمه الإبتدائية له داخل تركيبة for
- **int** i=3; تم تعريف العداد في الدالة الرئيسية وإعطاء القيمه الإبتدائية له داخل تركيبة for
  for(;i<=8;i--) {…….}
- **for(int** i=3;i<=8;) { i*=3;….} معدل التغير في العداد موضوع داخل جسم الحلقه
- **int** i=3; تم تعريف العداد وإعطاء القيمه الإبتدائية له في الدالة الرئيسية ووضع معدل التغير فيه في جسم الحلقه
  for(;i<=8;) { i/=3;….}
- **for(int** i=3,**int** j=9;i<=8||j>0;i=i+2,j--) {…….} حلقه مرتبطه بمتغيرين

مثال(3.5) اكتب برنامجاً بلغة C++ يطبع الأرقام الفردية odd number المحصورة بين 2 و 20

الحل: هذه الحلقه تحتاج لعداد يتغير بمعدل اثنين "الفرق بين كل عدد فردي و الذي يليه" و العداد قيمته الإبتدائية 3 وينتهي بـ 20 "شرط الحلقة" كما هو موضح في البرنامج:

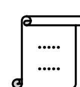


Prog7

```
#include<iostream.h>
void main()
{
for (int counter=3;counter<=20;counter+=2)
cout<<"\n"<<counter;
}
```

2- حلقة while

الصيغة العامة



while(condition)
{
. statements " loop body"
.
}

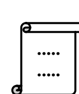
نفس الوضع القائم في الحلقة السابقة غير أن معدل التغير في قيمة العداد يكون داخل جسم الحلقه، وتعريف العداد وإعلان قيمته الإبتدائية قبل الحلقه، يمكن إعادة كتابة البرنامج السابق باستخدام حلقة **while:**

Prog8

```
#include<iostream.h>
void main()
{
      int counter=3;
while(counter<=20)
{
cout<<"\n"<<counter;
counter+=2;
}
```

3- حلقة do-while

الصيغة العامة



do
{
. statements " loop body"
.
}
while(condition);

نفس الوضع القائم في الحلقة السابقة غير أن معدل التغير في قيمة العداد يكون داخل جسم الحلقه، وتعريف العداد وإعلان قيمته الإبتدائية قبل الحلقه، نجد أن هذه الحلقه عكس حلقة **while** حيث اختبار الشرط مؤخر بعد جسم الحلقه مما يعني أن حلقة **do-while** لابد أن تنفذ مره على الأقل حتى ولو انتفى شرط الحلقه، يمكن إعادة كتابة البرنامج السابق باستخدام حلقة **do-while:**

Prog9

```
#include<iostream.h>
void main()
{
     int counter=3;
     do
       {
cout<<"\n"<<counter;
counter+=2;
}
while(counter<=20);
}
```

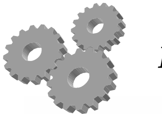
الحلقات المتداخله **Nested Loops**

حينما يتطلب الأمر البرمجي تكراره على مستويين تستخدم الحلقات المتداخله حيث تكتب حلقه داخل أخرى، مثلاً إذا أردنا التعامل مع درجات مجموعة طلاب في عدد من المواد فإن هنالك مستوي المواد والطلاب فيجعل لكل منهما حلقه خاصة، ونظراً لأن لكل طالب مجموعه من المواد فإن الحلقتين مرتبطتين ببعضهما البعض لذا توضع حلقه داخل أخرى. عدد مرات تنفيذ الأمر الموجود بالحلقة الداخلية يساوي حاصل ضرب عدد مرات التنفيذ لكل منهما.

مثال(3.6) اكتب البرنامج الذي يعطي الخرج التالي:

```
*********
*********
*********
*********
*********
*********
*********
*********
*********
```

الحل:



Prog10

```
#include<iostream.h>
   void main()
  {
     for (int  i=1;i<10;i++)
       {   //the code here will repeat 9 times
         for (int  j=1;j<10;j++)
           {
         cout<<"*"; //the code here will repeat 81 times
           }
         cout<<"\n"; //the code here will repeat 9 times
       }
  }
```

تمرين 3

1. كم عدد مرات تنفيذ الحلقات التالية:

a. **for**(i=1;i<15;i=i+3)
b. **for**(**int** i=0,**int** j=9;i<=16&&j>6;i++,j-=2)
c. int t=4;
while(t<=17)
{….
t+=3;}
d. int t=4;
do
{….
t+=3;} **while**(t>17);
e. **for**(i=9;i>5;i++)
f. **while**(6>3)
{……….}

2. أعد كتابة البرنامج *prog5* ليصلح لعدد n موظف؟

3. اكتب برنامجاً بلغة C++ يطبع الأعداد الأولية "primary numbers" في المدى من 10 و100

4. اكتب برنامجاً بلغة C++ يطلب إدخال درجة طالب في مادة ويخرج التقدير حسب الجدول التالي:

الدرجة "degree"	التقدير "grade"
95 فما فوق	A+
90-أقل من 95	A
85-أقل من 90	B+
80-أقل من 85	B
75-أقل من 80	C+
70-أقل من 75	C
65-أقل من 70	D+
60-أقل من 65	D
أقل من 60	F

5. أكتب برنامجاً بلغة C++ يحسب مضروب العدد "factorial" n! حسب العلاقه
n!=n*(n-1)*n-2)*……..3*2*1

6. أعد كتابة هذه الحلقة باستخدام **for** مره، ومره أخرى باستخدام **do-while** مره أخرى؟

```
int x=4;
while(x<10){
cout<<"\n C++";
x++;}
```

7. اكتب البرنامج الذي يعطي الخرج التالي:

أ-

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
```

ب-

```
* * * * * * * * * *
* * * * * * * * *
* * * * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

الفصل الرابع

int

float

المصفوفات



المصفوفات **Arrays**

المصفوفات هي أحد بنى البيانات الهامه، وهي عبارة عن مجموعه من خانات الذاكره المتتالية التي لها نفس الاسم ونفس نوع البيانات. ومن أجل الرجوع إلى خانه معينه من هذه الخانات نستخدم اسم المصفوفه ورقم العنصر في المصفوفه الذي يبدأ من 0.

الصيغه العامه للإعلان عن المصفوفه:

array_type array_name[elements_number];

مثلا الإعلان int x[5]; ، يمثل إعلان عن مصفوفة من الأعداد الصحيحه بها خمسه عناصر وتتميز عناصر المصفوفه بالرقم المحصور بين قوسي المصفوفه كالآتي:x[0],x[1],x[2],x[3],x[4].
يمكن أن تقرأ قيم عناصر المصفوفه عن طريق لوحة المفاتيح أو تسند لها القيم عن طريق معامل الإسناد"=" كما يلي:

array_type array_name[elements_number]={values};

يفترض أن يكون عدد العناصر مطابقا لعدد القيم وتسند القيم للعناصر على التوالي ابتداءً من العنصر الأول [0]، لكن عندما يكون عدد القيم أقل من العناصر تسند القيمة 0 تلقائياً للعناصر المتبقيه.

ملحوظه:

يمكن استخدام الحلقات التكراريه لتسهيل التعامل مع المصفوفات في عمليات الإدخال والإخراج أو المعالجة بحيث يستخدم عداد الحلقة للدلالة على رقم العنصر.

مثال (1.4) ***Prog11***

اكتب برنامجاً لحساب المجموع والمتوسط لعشرة أرقام مدخله؟

```
#include<iostream.h>
  void main()
   {
      float x[10],sum=0;
      cout<<"enter the numbers";
      for(int i=0;i<10;i++)
        {
          cin>>x[i];
          sum=sum+x[i];
        }
       float    average=sum/10;
      cout<<"the summation ="<<sum;
      cout<<"\nthe average ="<<average;
      }
```

ملحوظه:

تستخدم المصفوفات في تمثيل سلاسل الحروف كالأسماء حيث يكون نوع المصفوفه **char**.

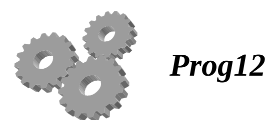
المصفوفات متعددة الأبعاد **Multiple Subscripts Array**

يمكن للمصفوفات أن تأخذ عدة أبعاد. ومن بين الاستخدامات الشائعة لهذا النوع من المصفوفات هو الجداول التي تنتظم فيها البيانات ضمن مجموعه من الصفوف و الأعمدة وبالتالي لنصل لمعلومه معينه يجب تحديد السطر والعمود وكل منهما يمثل بعداً للمصفوفه. المثال التالي يوضح الإعلان عن مصفوفه ذات بعدين:

float w[4][5]

فهذه المصفوفه بها 20 عنصراً "4*5" وللتعامل مع هذه المصفوفه في عمليات الإخراج والإدخال والمعالجه نستخدم الحلقات التكراريه التي مرت معنا سابقاً.

مثال(2.4) اكتب برنامجا يحسب متوسط درجات الطالب في أربعه مواد
الحل:



Prog12

```
//***this program to calculate the summation****
//***and the average of the student's degrees****
// ************in four subjects ***************
#include<iostream.h>
 void main()
 {
 char first_name[10],last_name[10];
 int degree[4];
 float sum=0,average;
 cout<<"\n enter the first name ";
 cin>>first_name;
 cout<<"\n enter the last name ";
  cin>>last_name;
 for(int k=0;k<4;k++)
 {
  cout<<"\n enter the degree ";
  cin>>degree[k];
  sum=sum+degree[k];
  }
  average=sum/4;
  cout<<" the degree average for ";
  cout<<first_name<<" "<<last_name;
  cout<<" is = "<<average;
 }
```

نلاحظ استخدام المصفوفات في تمثيل سلاسل الحروف في الاسم الأول والثاني للطالب في البرنامج السابق.

ملحوظه

- عند التعامل مع سلاسل الحروف ،يستخدم اسم المصفوفة الحرفية فقط .

مثال (3.4) اعد كتابة البرنامج السابق على أن يحسب متوسط الدرجات لعدد 6 طلاب كما في الجدول:

اسم الطالب	المواد				المتوسط
	فيزياء	رياضيات	احصاء	حاسب	
حسن علي	34	78	56	38	
علي سعد	67	90	89	90	
سالم علي	45	23	78	76	
صالح خليل	89	78	56	34	
فواز علي	45	78	63	54	
عمر خليل	47	58	78	63	

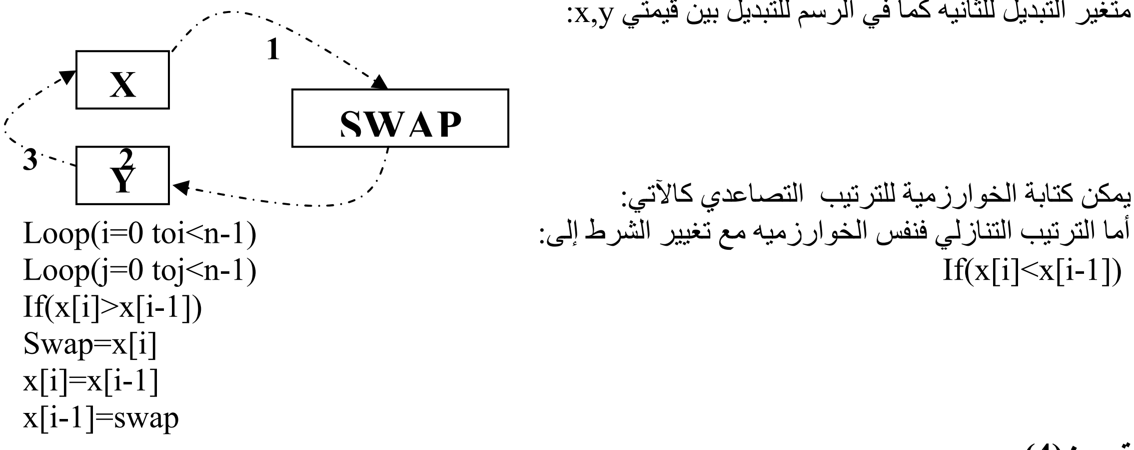
الجدول أعلاه يوضح درجات مجموعة طلاب في بعض المواد. لتمثيل هذه البيانات لابد لنا من التعامل مع بعدين هما الطلاب والمواد كما يظهر في البرنامج:

```
//***this program to calculate the summation**
//***and the average of the 6 students degrees
//  **************in four subjects ***************
#include<iostream.h>
  void main()
 {
 char first_name[10][6],last_name[10][6];
 int degree[4][6];
 float sum[6],average[6];
 for(int i=0;i<6;i++)
 {
 sum[i]=0;
 cout<<"\n enter the first name ";
 cin>>first_name[i];
 cout<<"\n enter the last name ";
  cin>>last_name[i];
 for(int k=0;k<4;k++)
 {
  cout<<"\n enter the degree ";
  cin>>degree[k][i];
  sum[i]=sum[i]+degree[k][i];
  }
  average[i]=sum[i]/4;
  }
  for(int j=0;j<6;j++){
  cout<<"\n the degree average for ";
  cout<<first_name[j]<<" "<<last_name[j];
  cout<<" is = "<<average[j];
  } }
```

نلاحظ استخدام مصفوفات ذات بعدين لتمثيل الدرجه degree و الإسم الأول first_name والإسم الأخير last_name.

استخدام المصفوفات في فرز البيانات:

يعتبر الفرز من أهم عمليات المعالجة المستخدمة في نظم التشغيل والبرامج التطبيقية، وتوجد عدة خوارزميات للفرز، منها ما سنتناوله الآن وهو الفرز الفقاعي bubble sort والذي يعتمد على مقارنه المتغيرات فيما بينها وتبديل مواقعها بما يتناسب مع نوع الترتيب سواءً كان تصاعدياً أم تنازلياً وهذا يتطلب مقارنة كل العناصر فيما بينها مما يستوجب استخدام حلقتين متداخلتين تعمل كل منهما n-1 مره حيث n عدد المتغيرات المراد ترتيبها.
عملية التبديل تتم باستخدام معامل الإسناد= حيث تسند القيمة الأولى لمتغير التبديل ثم القيمه الثانيه للأولى ثم قيمة متغير التبديل للثانيه كما في الرسم للتبديل بين قيمتي x,y:

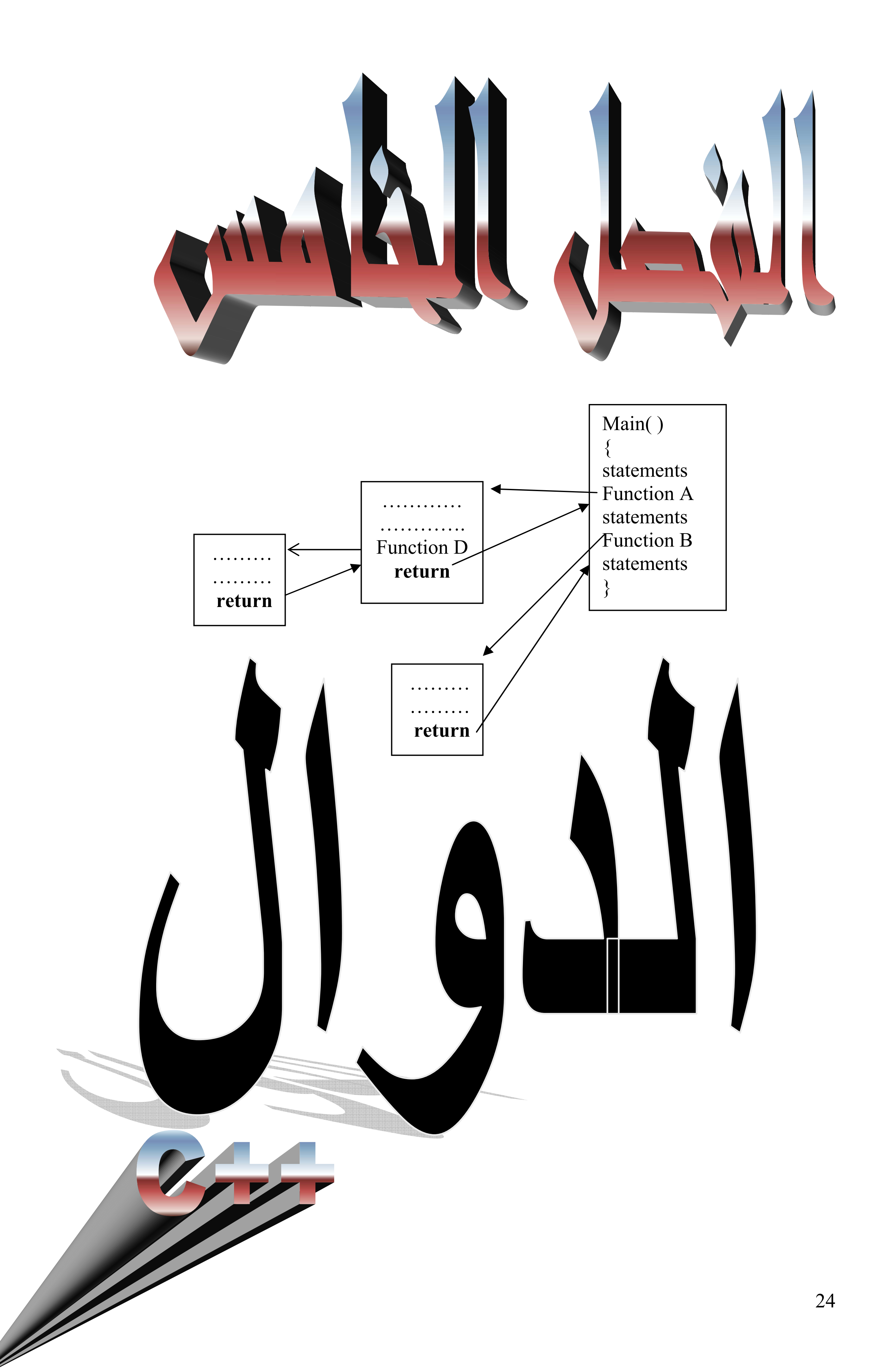



```
Loop(i=0 toi<n-1)
Loop(j=0 toj<n-1)
If(x[i]>x[i-1])
Swap=x[i]
x[i]=x[i-1]
x[i-1]=swap
```

يمكن كتابة الخوارزمية للترتيب التصاعدي كالآتي:
أما الترتيب التنازلي فنفس الخوارزميه مع تغيير الشرط إلى:
If(x[i]<x[i-1])

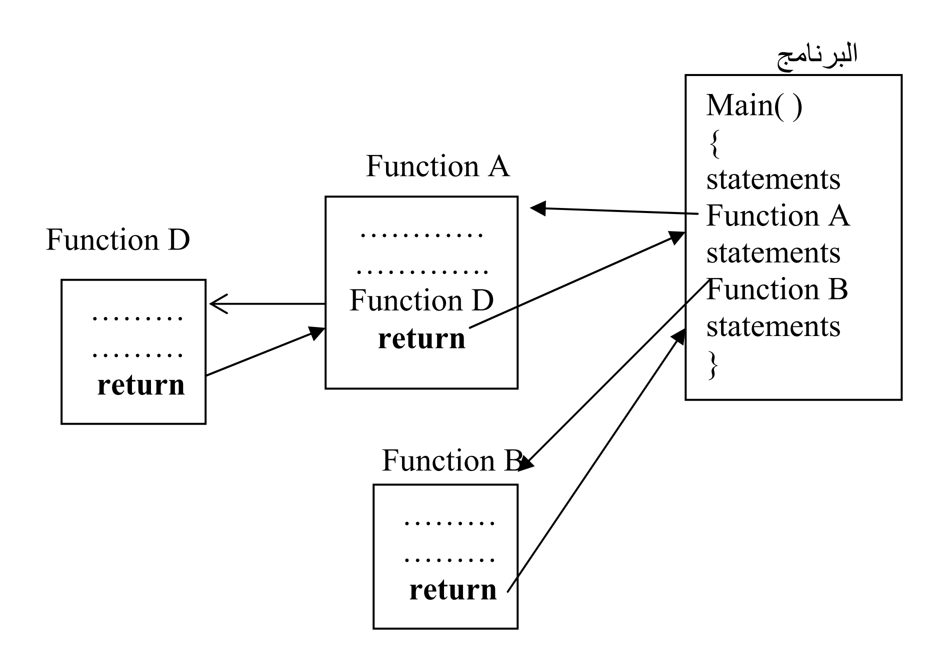
تمرين(4)

1. اكتب برنامج بلغة C++ يقوم بترتيب n رقم ترتيباً تصاعدياً ؟
2. أعد كتابة البرنامج 12 لعدد n طالب في m مادة بحيث يحسب متوسط درجات الطلاب في كل مادة؟


الفصل الخامس
Main( )
{
statements
Function A
statements
Function B
statements
}
…………
…………
Function D
return
………
………
return
………
………
return
الدوال
C++

الدوال Function

الدالة عباره عن برنامج فرعي يحتوي على تعليمه برمجيه أو أكثر ويتم تنفيذ الدالة عند استدعائها من قبل الدالة الرئيسيه أو أي داله أخرى ثم ترجع الداله قيمه للدالة التي قامت بالاستدعاء كما هو موضح بالرسم



تذخر المكتبه المعياريه للغه بمجموعه كبيره من الدوال الجاهزه والتي تتبع كل مجموعه منها لملف ترويسي لابد من ضمه للبرنامج عند الرغبه في استخدام احدى دوال هذا الملف، مثلاً الداله sqrt لحساب الجذر التربيعي لايمكن استخدامها الا إذاتم ضم الملف math.h للبرنامج، ويمكن الإطلاع على هذه الملفات و الدوال التي بداخلها من قائمة المساعده الموجوده في بيئة العمل وإليك بعض الدوال الرياضيه المنضويه تحت الملف math

الدالة	المهمة	الصيغة العامة	مثال
tan	حساب ظل الزاويه النصف قطريه	tan(x)	tan(0)=0
sin	حساب جيب الزاويه النصف قطريه	sin(x)	sin(0)=0
cos	حساب جيب تمام الزاويه النصف قطريه	cos(x)	cos(0)=1
abs	حساب القيمه المطلقه للأعداد الصحيحه	abs(y)	abs(-7)=7
ceil	تقريب لأعلى	ceil(t)	ceil(4.2)=5, ceil(-4.2)=-4
floor	تقريب لأسفل	floor(t)	floor(4.2)=4, floor(-4.2)=-5
pow	ناتج رفع عدد حقيقي لآخر	pow(x,y)	pow(2,3)=8, pow(.16,.5)=.4
sqrt	ايجاد الجذر التربيعي لعدد حقيقي	sqrt(w)	sqrt(9)=3, sqrt(.01)=.1
fmod	ايجاد باقي القسمه لعدد حقيقي	fmod(t,y)	fmod(5,3)=2,fmod(3,.7)=.2
log	ايجاد اللوغريثم الطبيعي لعدد حقيقي	log(x)	Log(4)=1.38629
Log10	ايجاد اللوغريثم العشري لعدد حقيقي	Log10(x)	Log10(100)=2

و هذه بعض الدوال المتعلقة بمعالجة سلاسل الحروف ضمن الملف string.h

الدالة	المهمة	الصيغة العامة	ملحوظة
strcpy	نسخ النص	strcpy(s1,s2)	نسخ s2في s1
strcat	دمج نص مع آخر	strcat(s1,s2)	تمثل s1 ناتج الدمج
Strlen	تحسب طول النص	strlen(s1)	ناتج الدالة عدد صحيح
strcmp	مقارنة نص مع آخر حسب ترميز آسكي	strcmp(s1,s2)	ترجع 0 إذا كان النصان متطابقان،1إذا كان ترميز s1أكبر من s2 -1إذا كان ترميز s1أصغر من s2
strrev	عكس النص	strrev(s1)	ترجع معكوس النص مثلا carتصبحrac

مثال(2.5)استخدم الدالة strrev في تشفير النص

C++ programming language is an object oriented language

Prog15

الحل

```
#include <string.h>
#include <iostream.h>
void main()
{
char text[]="C++ programming languge "
"is an object oriented language";
strrev(text);
cout<<"\n after reversing the text is: \n";
cout<<text;
}
```

الناتج

```
 after reversing the text is:
egaugnal detneiro tcejbo na si
 egugnal gnimmargorp ++C
```

إنشاء الدوال Function Creation

في الفقره السابقه تناولنا استخدام الدوال الجاهزه في المكتبه المعياريه للغه عن طريق استدعاء الداله، وكما هو واضح أن لكل داله صيغه خاصة لاستدعائها من حيث عدد الوسائطarguments فدالة sqrt تحتاج إلى وسيط واحد فقط هو العدد المراد إيجاد الجذر له، بينما نجد الدالةpow تحتاج إلى وسيطين الأساس والأس. ويمكن أن ينشئ المبرمج الدوال الخاصة به وبالتالي عليه الإعلان عن الداله وبنائها ومن ثم استدعائها، وهنا يجب تحديد:

- الأوامر والمهام التي ستقوم بها الداله.
- نوع وعدد الوسائط التي سترسل للداله.
- نوع القيم التي سترجعها الداله.

الإعلان عن الداله Function Declaration

المقصود بالإعلان تحديد مواصفات الداله من حيث نوع الإرجاعreturned type وعدد ونوع الوسائط arguments typeواسم الدالهfunction name.

الصيغه العامه:

Returned type function name(arg1 type name,arg2 type name,…….);

حيث arg1 type,arg2 type يمثلان نوع الوسيط "والوسيط هو متغير فيأخذ أحد أنواع المتغيرات المعروفه"، ويعقب ذلك name وهو اسم الوسيط وهو اختياري في الإعلان.

ملحوظه:

- يتم الإعلان عن الداله قبل بدء الداله الرئيسية
- إذا أسند للوسيط قيمه خلال الإعلان فستعتبر قيمه افتراضيه له ما لم ترسل للداله قيمه أخرى وهنا لا يلزم مناداة الداله بوسائط.

مثال(3.5)قم بالإعلان عن الدوال التاليه حسب المواصفات الموضحه في كل فقره مما يأتي:

أ. الداله swap لها وسيطين من نوع أعداد صحيحة وترجع عدد حقيقي؟
ب. الداله invert لها وسيط عدد صحيح طويل وترجع عدد صحيح طويل؟
ج. الداله print ليس لها وسائط ولا ترجع شئ؟

الحل:

أ. float swap(int, int)
ب. long int invert(long int)
ج. Void print()

تعريف الداله

تعريف الداله هو جسم الداله الذي يحتوي على الأوامر والتعليمات البرمجيه التي تنجز مهام الداله والذي يمكن أن يستخدم فيه أي من التعليمات البرمجيه التي درسناها آنفاً.

الصيغه العامه:

Returned type function name(arg1 type name,arg2 type name,…….)
{
function body
}

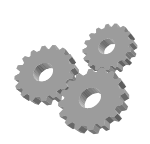
ملحوظات:

- إذا أعلن عن الداله أنها void فإن الداله لن ترجع شئ وإنما تنجز مهامها مباشرةً.
- إذا كانت الداله ترجع قيمه، لزم ذلك استخدام التعليمه return ملحقه بقيمة الإرجاع في آخر جسم الداله.
- جسم الداله يجب أن يكون مستقلاً عن الدوال الأخرى.

مثال(4.5) اكتب برنامجا يستخدم الدوال في تحديد القيمه الأكبر والقيمه الأصغر بين رقمين صحيحين مدخلين بحيث تسمى الدالتين max, min؟

الحل:

لو تمعنا في مهام الدالتين نجد أن كل منهما تحتاج إلى وسيطين "العددين المخلين" من نوع int وأن كل منهما ترجع عدداً صحيحاً أيضاً هو القيمه الأكبر في max أو الأصغر في min ويمكن استخدام المعامل الشرطي ?: في بنية الدوال لتحديد القيمه الأكبر أو الأصغر، لذا يكون البرنامج كالآتي:



Prog16

```
#include<iostream.h>
#include<math.h>
int max(int a,int b);
int min(int c,int d);
void main()
{
int x,y,max_number,min_number;
cout<<"\nenter the first number:";
cin>>x;
cout<<"\nenter the second number:";
cin>>y;
max_number=max(x,y);
min_number=min(x,y);
cout<<"\nthe mamimum number is ="<<max_number;
cout<<"\nthe minimum number is ="<<min_number;
}

int max(int e,int f)
 {
   int m=(e>f)?e:f;
   return m;
 }

 int min(int g,int h)
 {
   int n=(g<h)?g:h;
   return n;
 }
```

الإعلان عن الدوال (int max(int a,int b); و int min(int c,int d);)

استدعاء الدوال (max_number=max(x,y); و min_number=min(x,y);)

تعريف الدوال (int max(int e,int f) {…} و int min(int g,int h) {…})

إرسال المصفوفات كوسائط للداله:

وسيط الداله كما ذكرنا يمكن أن يكون متغير من أي نوع من أنواع البيانات المعروفه في C++، كما يمكن أن يكون مصفوفه من المتغيرات من أي نوع، فمثلا في المثال السابق لو أردنا أن نحصل على العدد الأكبر والأصغر من بين عشرة أعداد فإن مصفوفه من الأعداد الصحيحه سوف ترسل للداله.

ملحوظات:

- عند الإعلان والتعريف بالداله يستخدم اسم المصفوفه متبوعاً بأقواس المصفوفه خاليه.
- عند استدعاء الداله نكتفي باسم المصفوفه فقط.

يمكن إعادة كتابة البرنامج prog 16 لعشرة أرقام باستخدام المصفوفات كما يلي:

```
int max(int e[])
{
   int m=e[0];
   for(int w=0;w<10;w++)
   m=(m>e[w])?m:e[w];
   return m;
}
int min(int g[])
{
   int n=g[0];
   for(int d=0;d<10;d++)
   n=(n<g[d])?n:g[d];
   return n;
}
#include<iostream.h>
void main()
{
int x[10],max_number,min_number,counter=0;;
cout<<"\nenter the  numbers:";
while(counter<10){
cin>>x[counter];
counter++;}
max_number=max(x);
min_number=min(x);
cout<<"\nthe mamimum number is ="<<max_number;
cout<<"\nthe minimum number is ="<<min_number;
}
```

الدوال العوديه Recursive Function

من الملاحظ أن نداء الدالة يتم من نقطه في البرنامج خارج الداله، وهي بدورها تقوم بإرجاع قيمه لنفس النقطه التي تم فيها النداء وقد يتطلب البرنامج أن تكرر الداله نفسها عدة مرات وبالتالي تقوم الداله باستدعاء نفسها وهذا ما يعرف بالدوال العوديه.

ملحوظه :

- يستخدم الأسلوب العودي عندما تكون الداله علاقة في نفسها أي على الصيغه:د(س)=د(د(س))
مثلاً د(س)=د(س)+د(س+1)

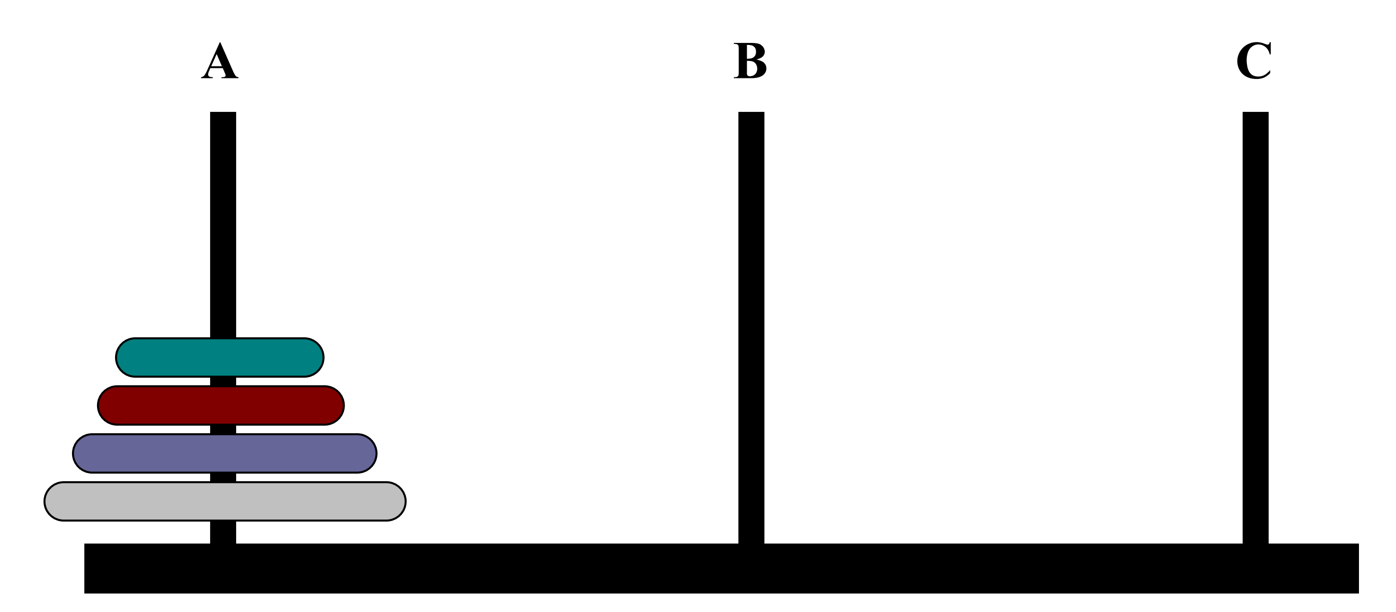
مثال(5.5) الصيغ التالية تمثل أشكالاً لدوال عوديه:

- **int** xx(**int** a)
  {
  …
  ….
  return xx(a)
  }
- **float** ww(**int** b);
  {
  ….
  ….
  return ww(b)+ww(b-1);

مشكلة أبراج هانوي **Hanoi Tower**

يستعمل هذا المثال لتوضيح الدوال العوديه والمشكلة تتلخص في وجود ثلاثة أعمدة A, B, C ويوجد عدد من الأقراص متدرجة القطر على العمود Aبحيث يكون القرص ذو القطر الأكبر أسفل، و الذي أقل منه أعلى، كما هو موضح بالشكل، والمطلوب نقل الأقراص إلى العمود C بنفس الوضع بالإستعانه بالعمود B مع مراعاة عدم وضع قرص على آخر أقل منه في القطر.

المطلوب برنامج يحدد خطوات الحل حسب عدد الأقراص المدخل؟



الحل:

لو حاولنا حل المسألة في حالة وجود قرصين نجد الآتي:

حرك قرص من A إلى B.

حرك قرص من A إلى C.

حرك قرص من B إلى C.

إذن يتم الحل في 3 خطوات، أما إذا كان عدد الأقراص 3 فإننا نحتاج لنقل القرصين العلويين من A إلى B " وهذا يحتاج إلى 3 خطوات" ثم ينقل القرص الأخير من A إلى B في خطوه واحده، ثم ينقل القرصين العلويين من B إلى C " وهذا يحتاج إلى 3 خطوات"، إذن يتم الحل في 7 خطوات، يمكن تلخيص خطوات الحل كالآتي:

- أنقل n-1 قرص من A إلى B
- انقل القرص الأخير من Aإلى C
- أنقل n-1 قرص من B إلى C

نجد أن في الخطوتين الأولى والثالثة تطبيق لمبدأ العوديه حيث يتم استدعاء الداله من داخلها.

نجد أن عدد خطوات الحل يساوي عدد خطوات الحل لـ n-1 قرص مضروباً في اثنين زائداً واحد أي أن

د(ن)=2*د(ن-1)+1

مثلاً إذا ن=5 فإن

د(5)=2*د(4)+1 وهذا لايمكن الحصول عليه إلا إذا حسبنا د(4)

د(4)=2*د(3)+1

د(3)=2*د(2)+1

د(2)=2*د(1)+1

د(1)=1

إذن د(5)=31 إذن هذه الداله تحتاج إلى تطبيق لمبدأ العوديه، البرنامج التالي يوضح حل هذه المشكلة:

Prog18

```
#include<iostream.h>      //hannoy tower
void move_tower(int,char,char,char);
int count(int);
void main() {
    int n;
    char source='A',distination='C',spare='B';
    cout<<"\n enter the no of dish ";
    cin>>n;
 move_tower(n,source,distination,spare);
 cout<<" \n the problem is completed in "<<count(n)<<" steps";}

void move_tower(int n,char source,char distination,char spare) {
if(n==1)
cout<<"\nmove a disk from peg "<<source<<" to peg "<<distination;
else{
 move_tower(n-1,source,spare,distination); //recurssion
 cout<<"\nmove a disk from peg "<<source<<" to peg "<<distination;
 move_tower(n-1,spare,distination,source); //recurssion
 } }

 int count(int a){
   if(a==1)
   return 1;
   else
    return 1+2*count(a-1); //recurssion
    }
```

مثال(6.5)اكتب برنامجاً يحسب الحد n من سلسلة فايبوناتشي Fibonacci Series التي تمثل الأرقام 0,1,1,2,3,5,8,13,21,…………

حيث الحد العام ح(ن) يحسب بالعلاقة:

ح(ن) =ح(ن-1)+ ح(ن+2) مع العلم أن ح(0)=0،ح(1)=1

الحل:النسبة بين أي حد والذي يسبقه تسمى بالنسبه الذهبيه *golden ratio* وتستخدم في التصميم المعماري كنسبه بين الطول والعرض للنوافذ والغرف، من صيغة الحد العام لهذه السلسله نجد أن أي حد من حدودها هو داله في الحدين السابقين له "حاصل جمعهما" أي أن العلاقه داله في نفسها، لذا يستحسن أن تكون الدالة Fibonacci داله عوديه، يبدأ البرنامج بطلب إدخال ترتيب الحد المراد إيجاد قيمته وفي حال إدخال قيمه 0 أو1 فإن الدالة ترجع n.

Prog19

```
#include<iostream.h>
int fibonacci(int);
void main(){
     int n;
     cout<<"\n enter the item order: ";
     cin>>n;
     cout<<"\nfibonacci value is: "<<fibonacci(n);
  }
  int fibonacci(int number){
      if(number==0||number==1)
      return number;
      else
      return fibonacci(number-1)+fibonacci(number-2);
   }
```

فرط تحميل الدوال Function Overloading

المقصود بذلك تعدد الدوال الشكلي حيث يمكن أن تعطي أكثر من داله الاسم نفسه مع تغيير في عدد الوسائط أو نوعها ويرسل استدعاء الداله بعد مطابقة عدد ونوع الوسائط المثال التالي يمثل فرط تحميل الداله dd حيث تظهر مره بوسيط واحد من نوع int، ومره بوسيط من نوع double ، ومره بدون وسيط:

Prog20

```
#include<iostream.h>
int dd(int);
void dd();
double dd(double);
void main()
{
cout<<dd(3);
cout<<dd(1.2);
dd();
}
int dd(int a) {
cout<<"\n from int 3*3= ";
return a*a;
}
    void dd()
  {cout<<"\n from void just text";}

  double dd(double b){
  cout<<"\n from double 1.2*1.2= ";
  return b*b;
  }
```

الدوال المباشرة Inline Function

عند تعريف دالة ما تنشأ نسخه من تعليمات هذه الداله في الذاكرة وعند استدعاء الداله ينتقل تنفيذ البرنامج إلى هذه التعليمات ثم يعود إلى أول سطر بعد الاستدعاء، فإذا استدعيت داله عدد من المرات فإن وقت تنفيذ البرنامج سيهدر بسبب القفز من وإلى الداله، وللتقليل من هذا الأثر يعلن عن الداله أنها خطيه عن طريق الكلمة المحجوزة inline مما يؤدي إلى التخلص من القفز بأن تنشأ نسخه من الداله inline إلى الداله المستدعية.

ملحوظه:

- ليس من المناسب استخدام inline إذا كان حجمها كبيرا وعدد مرات الاستدعاء للداله كثير لأن ذلك سيؤدي إلى نسخ الداله inline عدة مرات مما يؤدي لتضخم المساحة المحجوزة للبرنامج وبالتالي بطء في أداء البرنامج.

المتغيرات المحلية والمتغيرات الشاملة Global & Local Variables

المتغيرات المعرفه خارج نطاق أي داله تسمى متغيرات شامله و تكون متاحه لأي داله في البرنامج بما في ذلك الداله الرئيسيه، أما المتغيرات المعرفه داخل نطاق أي داله فهي متغيرات محليه لا يمكن استخدامها إلا في نطاق الداله نفسها. البرنامج التالي يوضح هذا المفهوم حيث يمثل a متغيراً شاملاً يمكن تناوله من أي داله، غير أن y متغير محلي معرف للداله الرئيسية فقط.

Prog21

```
#include<iostream.h>
void f1();
int a=7;  //global variable
void main()
{
int y=3;  //local variable
f1();
cout<<"\n y="<<y;
cout<<"\n from main a="<<a;
}
void f1()
{cout<<"\n from f1 a="<<a;}
```

إنشاء ملفات الترويسه الخاصه بالمبرمج

كما مر معنا سابقاً وجود عدد من ملفات الترويسهHeader Files في مكتبة اللغة المعياريه وبإمكان المبرمج انشاء ملفات خاصه به عن طريق حفظ الملف بالإمتدادh ويتم ضم الملف لأي برنامج عن طريق الصيغه#include"file_name"

تمرين(5)

1. ما هو ناتج التعليمات التاليه:
   - أ. cout<<strlen("I love C++ programming")
   - ب. cout<<strrev("I love C++ programming")
   - ج. strcat("death to ","israil")
   - د. pow(4,2)
   - ه. sqrt(.36)
   - و. ceil(2.05)
   - ز. floor(3.99)
   - ح. ceil(-8.2)
   - ط. floor(-.99)
2. أعلن عن الدوال التاليه:
   - أ. الداله aa لها وسيط واحد حرفي ولا ترجع شئ.
   - ب. الدالهbb لها وسيط عدد حقيقي وترجع عدد صحيح.
3. استخدم الدوال العوديه لحساب مضروب العدد؟